Universal Semantic Parsing: Supplementary Material

Siva Reddy† Oscar Täckström†† Slav Petrov†† Mark Steedman†† Mirella Lapata††
† Stanford University
†† Google Inc.

sivar@stanford.edu, {oscart, slav}@google.com, {steedman, mlap}@inf.ed.ac.uk

Abstract

This supplementary material to the main paper provides an outline of how quantification can be incorporated in the UDEP-LAMBDA framework.

1 Universal Quantification

Consider the sentence Everybody wants to buy a house,1 whose dependency tree in the Universal Dependencies (UD) formalism is shown in Figure 1(a). This sentence has two possible readings: either (1) every person wants to buy a different house; or (2) every person wants to buy the same house. The two interpretations correspond to the following logical forms:

(1) ∀x. person(x) →
   [∃y. wants(y) ∧ arg1(zr, x) ∧ buy(y) ∧ xcomp(zr, y) ∧
    house(w0) ∧ arg1(zr, x) ∧ arg2(zr, w0)] ;

(2) ∃y. house(w0) ∧ (∀x. person(x) →
   [∃y. wants(y) ∧ arg1(zr, x) ∧ buy(y) ∧ xcomp(zr, y) ∧
    arg1(zr, x) ∧ arg2(zr, w0)]).

In (1), the existential variable w is in the scope of the universal variable x (i.e. the house is dependent on the person). This reading is commonly referred to as the surface reading. Conversely, in (2) the universal variable x is in the scope of the existential variable w (i.e. the house is independent of the person). This reading is also called inverse reading.

Our goal is to obtain the surface reading logical form in (1) with UDEP-LAMBDA. We do not aim to obtain the inverse reading; although this is possible with the use of Skolemization (Steedman, 2012).

In UDEP-LAMBDA, lambda expressions for words, phrases and sentences are all of the form λx…λz. But from (1), it is clear that we need to express variables bound by quantifiers, e.g. ∀x, while still providing access to x for composition. This demands a change in the type system since the

1Example borrowed from Schuster and Manning (2016).

Figure 1: The dependency tree for Everybody wants to buy a house and its enhanced variants.

same variable cannot be lambda bound and quantifier bound—that is we cannot have formulas of the form λx…∀x… In this material, we first derive the logical form for the example sentence using the type system from our main paper (Section 1.1) and show that it fails to handle universal quantification. We then modify the type system slightly to allow derivation of the desired surface reading logical form (Section 1.2). This modified type system is a strict generalization of the original type system.2 Fancellu et al. (2017) present an elaborate discussion on the modified type system, and how it can handle negation scope and its interaction with universal quantifiers.

2Note that this treatment has yet to be added to our implementation, which can be found at https://github.com/sivareddyg/udeplambda.
1.1 With Original Type System

We will first attempt to derive the logical form in (1) using the default type system of UDEPLAMBDA. Figure 1(b) shows the enhanced dependency tree for the sentence, where BIND has been introduced to connect the implied nsubj of buy (BIND is explained in the main paper in Section 3.2). The s-expression corresponding to the enhanced tree is:

\[
(\text{nsubj} \ (\text{xcomp} \ \text{wants} \ (\text{mark} \\
\text{nsubj} \ (\text{dobj} \ \text{buy} \ (\text{det} \ \text{house} \ a) \ \Omega) \to)) \\
(\text{BIND} \ \text{everybody} \ \Omega))
\]

With the following substitution entries,

\[
\text{wants} , \ \text{buy} \in \text{EVENT}; \\
\text{everybody} , \ \text{house} \in \text{ENTITY}; \\
a , \ \text{to} \in \text{FUNCTIONAL}; \\
\Omega = \lambda \times \text{EQ}(x, \Omega); \\
\text{nsubj} = \lambda f . g . x . y . f(x) \land g(y) \land \text{arg}_1(x, y, a); \\
\text{dobj} = \lambda f . g . x . y . f(x) \land g(y) \land \text{arg}_2(x, y, a); \\
\text{xcomp} = \lambda f . g . x . y . f(x) \land g(y) \land \text{xcomp}(x, y, a); \\
\text{mark} \in \text{HEAD}; \\
\text{BIND} \in \text{MERGE};
\]

the lambda expression after composition becomes:

\[
\lambda \cdot \exists x y w : \text{wants}(z_e) \land \text{everybody}(x_a) \land \text{arg}_1(z_e, x_a) \\
\land \text{EQ}(x, \Omega) \land \text{buy}(y_1) \land \text{xcomp}(z_e, y_e) \land \text{arg}_1(y_e, y_1) \\
\land \text{EQ}(y, \Omega) \land \text{arg}_1(x_e, y_a) \land \text{house}(w_a) \land \text{arg}_2(y_e, w_a).
\]

This expression encodes the fact that \( x \) and \( y \) are in unification, and can thus be further simplified to:

\[
(3) \ \lambda \cdot \exists x y : \text{wants}(z_e) \land \text{everybody}(x_a) \land \text{arg}_1(z_e, x_a) \\
\land \text{buy}(y_1) \land \text{xcomp}(z_e, y_e) \land \text{arg}_1(y_e, x_a) \\
\land \text{arg}_1(x_e, y_a) \land \text{house}(w_a) \land \text{arg}_2(y_e, w_a).
\]

However, the logical form (3) differs from the desired form (1). As noted above, UDEPLAMBDA with its default type, where each s-expression must have the type \( \eta = \text{Ind} \times \text{Event} \to \text{Bool} \), cannot handle quantifier dosing.

1.2 With Higher-order Type System

Following Champollion (2010), we make a slight modification to the type system. Instead of using expressions of the form \( \lambda x \ldots \) for words, we use either \( \lambda f . \exists x \ldots \) or \( \lambda f . \forall x \ldots \), where \( f \) has type \( \eta \). As argued by Champollion, this higher-order form makes quantification and negation handling sound and simpler in Neo-Davidsonian event semantics. Following this change, we assign the following lambda expressions to the words in our example sentence:

\[
everybody = \lambda f . \forall x . \text{person}(x) \to f(x);
\]
\[
wants = \lambda f . \exists x . \text{wants}(x) \land f(x);
\]
\[
to = \lambda f . \text{TRUE};
\]
\[
buy = \lambda f . \exists x . \text{buy}(x) \land f(x);
\]
\[
a = \lambda f . \text{TRUE};
\]
\[
house = \lambda f . \exists x . \text{house}(x) \land f(x);
\]
\[
\Omega = \lambda f . f(\Omega).
\]

Here everybody is assigned universal quantifier semantics. Since the UD representation does not distinguish quantifiers, we need to rely on a small (language-specific) lexicon to identify these. To encode quantification scope, we enhance the label nsubj to nsubj:univ, which indicates that the subject argument of wants contains a universal quantifier, as shown in Figure 1(c).

This change of semantic type for words and s-expressions forces us to also modify the semantic type of dependency labels, in order to obey the single-type constraint of DEPLAMBDA (Reddy et al., 2016). Thus, dependency labels will now take the form \( \lambda P Q f \ldots \), where \( P \) is the parent expression, \( Q \) is the child expression, and the return expression is of the form \( \lambda f . . . \). Following this change, we assign the following lambda expressions to dependency labels:

\[
\text{nsubj:univ} = \lambda P Q f . Q(\lambda y . P(\lambda x . f(x) \land \text{arg}_1(x, y, a)));
\]
\[
\text{nsubj} = \lambda P Q f . P(\lambda x . f(x) \land Q(\lambda y . \text{arg}_1(x, y, a)));
\]
\[
\text{dobj} = \lambda P Q f . P(\lambda x . f(x) \land Q(\lambda y . \text{arg}_2(x, y, a)));
\]
\[
\text{xcomp} = \lambda P Q f . P(\lambda x . f(x) \land Q(\lambda y . \text{arg}_3(x, y, a)));
\]
\[
\text{mark} = \lambda P Q f . P(\lambda x . f(x) \land Q(\lambda y . \text{EQ}(x, y)));
\]

Notice that the lambda expression of nsubj:univ differs from nsubj. In the former, the lambda variables inside \( Q \) have wider scope over the variables in \( P \) (i.e. the universal quantifier variable of everybody has scope over the event variable of wants) contrary to the latter.

The new s-expression for Figure 1(c) is

\[
(\text{nsubj:univ} \ (\text{xcomp} \ \text{wants} \ (\text{mark} \\
(\text{nsubj} \ (\text{dobj} \ \text{buy} \ (\text{det} \ \text{house} \ a) \ \Omega) \to)) \\
(\text{BIND} \ \text{everybody} \ \Omega)).
\]

Substituting the modified expressions, and performing composition and simplification leads to the expression:

\[
(6) \ \lambda f . \forall x . \text{person}(x) \to \exists y w : \text{wants}(z_e) \land \text{arg}_1(z_e, x_a) \land \text{buy}(y_e) \\
\land \text{xcomp}(z_e, y_e) \land \text{house}(w_a) \land \text{arg}_1(z_e, x_a) \land \text{arg}_2(z_e, w_a)).
\]

This expression is identical to (1) except for the outermost term \( \lambda f \). By applying (6) to \( \lambda f . \text{TRUE} \), we obtain (1), which completes the treatment of universal quantification in UDEPLAMBDA.

References


